



# **Army Enterprise Integration Oversight Office (AEIOO)**

## **Interfaces for Enterprise Solutions**

**January 2005**

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

### Executive Summary

In virtually all sectors, management is seeking increased value from their organization's business processes. Moving information through the enterprise (internally and externally), from person to person, in as little time as possible facilitates streamlining business processes and reducing costs. This minimal transfer time or zero-latency enterprise concept is a key driver for the current popularity of integrating disparate applications across the extended enterprise. In large, complex organizations, the extended enterprise is far-reaching, representing domestic and overseas operations, as well as business partners. This flow of information is generically referred to as an 'interface' in this document. However, an interface can take on many forms, from simply exchanging basic data between two applications to exchanging data and complex business logic within an unlimited network of remote systems. This document discusses these concepts by providing information on fundamental interface concepts, interface models, and interface cost analysis.

Fully integrating an organization under one enterprise system may eliminate the need for interfaces, but this is not feasible in most large enterprises. Management must evaluate and implement a coordinated interface model, preferably before systems are implemented. This interface model is aligned with the organization's existing enterprise architecture. Performing these steps avoids fostering the 'accidental architecture', which exists where multiple incompatible interface solutions are adopted over time, creating 'islands of integration'. With interface technology changing so rapidly, organizations are struggling to select (and maintain) an interface model that meets their needs. Selecting an interface model that adopts widely-used standards, such as the eXensible Markup Language (XML), will help ensure compatibility with the plethora of software products. Exchanging information within the extended enterprise requires a shared communication language, one that standards greatly facilitate. When evaluating standards, organizations should realize the value of commercial standards over industry or organization specific standards because of their widespread applicability and 'out-of-the-box' software compatibility.

While traditional interface models, such as point-to-point and hub and spoke, have been widely deployed for many years, they are costly to maintain and do not provide the loose-coupling and fast return-on-investment that many Chief Information Officers are now seeking. A variant of the hub and spoke model being implemented by Enterprise Resource Planning (ERP) system vendors provides a step in the right direction with pre-configured integration and adaptability for outlying applications. Web services and the Enterprise Service Bus (ESB) model are promising interface technologies that provide access to loosely coupled services using commercial standards and existing infrastructure. Web services and ESB support a service oriented architecture for either an organization's internal applications or those integrated with the extended enterprise.

Building interfaces is time consuming and expensive. Interface work can represent up to 40 percent of total rollout costs for large-scale applications.[1] Interface costing is analogous to purchasing a car – while the base price may provide an initial guess of the cost, it isn't in the ballpark until options, fees, and long-term value are considered. One million dollars is likely an average estimate for the cost of a complex interface in a large enterprise; however, many factors such as the interface model, the cost estimating method, and functional, technical, and organizational characteristics significantly impact the cost of an interface. By having an understanding of interface technology and cost factors, system architects are better equipped to design and deploy cost-effective interface models.

#### *Special Thanks To:*

*Scott Bernard, Ph.D.  
John Browne, Jr.  
Tom Gulledge, Ph.D.  
Jon Marshall  
Amy Taylor  
Dr. Paul Tibbits*

*Carla von Bernewitz  
Paul Derby, Ph.D.  
Alan Jones  
John Nyere  
Jess Thompson  
Larry Wright*

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

### Table of Contents

<u>Section</u>	<u>Page</u>
A. Introduction .....	1
Purpose & Scope	
Glossary of Terms, Acronyms & References	
Evolution of Interfaces	
Objectives & Benefits of Interfaces	
B. Interface Concepts .....	4
Interface Technology Fundamentals	
Standards	
Interface Documentation	
C. Interface Models .....	7
D. Interface Cost Analysis .....	9
Interface Cost Estimating Methods	
Counting Interfaces	
E. The Road Ahead .....	13
F. Wrap-up .....	14
Appendix A – Interface Model Descriptions.....	App. A
Appendix B - Glossary of Terms.....	App. B
Appendix C – Acronyms .....	App. C
Appendix D - References .....	App. D

### List of Figures & Tables

Figure 1 - Basic Interface.....	4
Figure 2 - Characteristics of a Successful Interface .....	5
Table 1 - Interface Model Characteristics .....	8
Figure 3 - Lifecycle of Costing Methods.....	10
Table 2 - Interface Cost Estimate .....	12
Table 3 - Interface Characteristics .....	13
Figure 4 - Point-2-Point Interface.....	App. A.1
Figure 5 - Hub and Spoke Interface.....	App. A.2
Figure 6 - Web Services Process Flow.....	App. A.4
Figure 7 - Web Services Interface .....	App. A.5
Figure 8 - The Accidental Architecture.....	App. A.6

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

### A. Introduction

*"Possibly the single most transforming thing in our forces will not be a weapons system, but a set of interconnections and a substantially enhanced capability because of that awareness."*

Secretary of Defense  
Donald Rumsfeld  
August 9, 2001 [2]

In the Industrial Age the interconnections Mr. Rumsfeld refers to may have consisted of verbal agreements between business partners. However, in the Digital Age these interconnections take the form of a system interface.

An interface can be defined as a physical or logical connection between two systems in order to communicate or work together effectively. Similarly, the Department of Defense's (DoD) RICE (Reports, Interfaces, Conversions, Extensions) Development Process and Procedures document defines an interface as "a boundary across which two independent systems meet and act on or communicate with each other." [3]

The late '90s ushered in the popularity of Enterprise Resource Planning (ERP) packages such as SAP, PeopleSoft and Oracle Applications. This was partly due to the idea of integrating various business processes with a single system to allow near real-time access to relevant information. Thus, efficiency and effectiveness were gained by cross-functional process integration. For a variety of reasons, organizations that migrated to an ERP were left with outlying processes that were not supported by the enterprise system. To integrate these outlying processes (and their supporting systems) with the enterprise system, interfaces were implemented to share information between the ERP and the outlying system. Whether using an ERP, best-of-breed approach, in-house approach, or a mix of these approaches, organizations, such as the United States Army, spend millions of dollars every year building and maintaining interfaces between systems. Integration among new and existing applications is a problem that costs companies over \$100 billion per year. Often, application interfaces can grow more complex than the applications themselves. [4]

Having recognized there is a need to integrate, enterprises should design an interface model before implementing the first interface. But as simple as this may seem, the design carries consequences that can have extraordinary impacts on the organization. A point-to-point approach may be fast to implement and technically simple, but it may cheat the enterprise out of multiple points of efficiency. A hub and spoke approach, such as Enterprise Application Integration (EAI), may be overly complex and feature-filled, but ultimately miss the mark of the enterprise's needs. [5]

Although the focus of this document is on system interfaces, interfaces, integration and interoperability are closely related. Paramount to the interface topic is the idea of an enterprise architecture and the requirements of business processes. First, it is absolutely essential that an enterprise architecture be in place to guide the selection of an interface model. For instance, Federal agencies are obligated to align with the Federal Enterprise Architecture (FEA), which is a set of reference models that promote cross-agency integration and collaboration. Second, business requirements should drive IT solutions [6], including interfaces. The E-Gov initiatives are excellent examples of business requirements for interfacing data between systems (e.g., E-Travel, E-Clearance, E-Grants, for more see: [www.whitehouse.gov/omb/egov/c-presidential.html](http://www.whitehouse.gov/omb/egov/c-presidential.html)). This document assumes that the interface model will align with an enterprise architecture and the business requirements will drive the design of the interface.

While most system integrators agree there are various degrees of interface sophistication, few agree on the terms used to describe them. In this document, the term *interface* will be used generically to

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

refer to any type of data/information exchange between systems, regardless of the degree of integration or interoperability. The terms describing integration are used as follows in this document:

- *Interconnecting* - this is the lowest level of interface sophistication, described as an antiquated style of interfacing flat file data.
- *Integration* – this is defined as the next level of sophistication where an enterprise's business processes can operate collectively in a seamless, event-driven fashion because the underlying systems communicate with each other through the transfer of data and semantics.
- *Interoperability* – As opposed to the two above, interoperability is more a characteristic that objects (e.g., systems, enterprises) can possess rather than perform. From a data standpoint, it describes the ability of systems to exchange data without requiring transformation, because the systems have adopted the same standards. However, from a broader perspective, it describes the ability of objects to 'work together' with a shared ontology (i.e., each party understanding the semantics, context, etc. of messages exchanged).

DoD defines many more levels of information-exchange sophistication in the LISI (Levels of Information Systems Interoperability) Maturity Model. The LISI Model's five levels of interoperability include Enterprise, Domain, Functional, Connected, and Isolated. For more information, refer to [www.sei.cmu.edu/isis/presentations/sstc-incose/sld016.htm](http://www.sei.cmu.edu/isis/presentations/sstc-incose/sld016.htm).

### Purpose & Scope

For many years to come interfaces will be the glue that connects disparate business processes, enabling applications to exchange multi-layered information. However, managing interfaces involves first defining the interface requirements and then evaluating alternative interface models.[7] The second step is what this document focuses on by analyzing predominant and emerging interface models. Guidance on how to determine the most cost-effective alternative interface model is also discussed.

This document provides fundamental information regarding software interface concepts, interface models, and interface cost analysis. The scope includes the technology and financial aspects surrounding software interfaces for information systems. Within this document a software interface includes any logical interaction and transmission of information between two information systems using a program or multiple programs. The scope does not include user interfaces or hardware interfaces.

### Glossary of Terms

Refer to **Appendix B** for the Glossary of Terms.

### Acronyms

Refer to **Appendix C** for the list of acronyms.

### References

Refer to **Appendix D** for document references shown in brackets [ ].

# **Interfaces for Enterprise Solutions**

## **Army Enterprise Integration Oversight Office (AEIOO)**

---

### **Evolution of Interfaces**

Historically, interfaces were confined to the technical aspects of hardware and the interconnectivity of computing components. Interfaces had a mechanical connotation and piecemeal quality: making different pieces of equipment work together. As the computing industry and knowledge evolved, interfaces began to include software, data and communication. The goal became transferring information from one system to another electronically to avoid reentering it manually. This was initially widely deployed using a batch approach where a set of information was transferred on a periodic basis. Then as software and hardware advanced, transferring information in real-time became the norm and individual transactions could be transmitted instead of a batch of transactions.

Electronic Data Interchange (EDI) became a popular method to interface with other organizations by using an intermediary to facilitate the data exchange. However, high costs and EDI's proprietary technology prevented its widespread use. In the 1990's the desire for organizations to exchange information internally and externally grew with the pace of globalization and competition.

Organizations sought ways to exchange more of their information and exchange it faster. However, grand ideas about the Web blurred organizations' focus on the value of interfacing information both internally and externally. Business demands and new technology in the areas of middleware, standard protocols, and software tools helped organizations refocus on the value of interfacing and integrating their systems.

The focus has moved from simply interfacing within organizations to integration within and among organizations. Although outside the scope of this document, integration has come to mean more than just technology. Systems integration involves a complete system of business processes, managerial practices, organizational interactions, structural alignments, and knowledge management. It is an all-inclusive process designed to create seamless and highly agile processes and organizational structures that are aligned with the strategic and financial objectives of the enterprise. System integration is achieved when the processing environment, technologies, human performance, and business processes all function in a harmonious and congruent manner.[8] This full blown concept of integration has not yet become ubiquitous, but this document provides insights on how to get there.

### **Objectives & Benefits of Interfaces**

The ultimate objectives of interfaces are to reduce costs and improve efficiency, effectiveness, responsiveness, and stakeholder satisfaction. The underlying objectives of interfacing systems contribute to achieving these ultimate objectives. For instance, overall costs can be reduced by eliminating manual data entry while operating efficiencies can be gained by interfacing data quickly between systems. Other underlying objectives of interfaces include increased access to information, improved interoperability, enhanced self-service capabilities, reduction of errors, increased growth capability, and the most important – business process alignment and integration.

Without delving into the differences between interfacing and integration, it is important to at least understand that the concept of event-driven sharing of information among systems is a core component of the Net-Centric approach. Modern interface models can provide efficient and effective communication between systems that enable the attainment of net-centricity. Delving down from the process level to the data level, interfaces/integration provides the connectivity that allows communities of interest to exist. A community of interest is a network that crosses organizational and service boundaries to provide disparate systems access to shared data.

Increased collaboration and coordination is a more recent objective of system interfacing, commonly referred to as integration. New technologies such as Web services and standardized Internet protocols have transformed the way people collaborate to exchange information and develop solutions. Further, these technology capabilities are available around the clock as opposed to the

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

“business hours” of the Industrial Age. New technologies have transformed the way organizations interact with employees, suppliers, and customers. No longer does one need to wait until the bank opens in the morning to make a transfer between accounts – it can be done on-line around the clock. With an enterprise such as the U.S. Army requiring interaction and data exchange across all time zones, the ability to conduct operations regardless of location, time of day, or personnel availability has obvious advantages.

Although interfaces provide many advantages, organizations continually fall prey to their pitfalls. Organizations that do not select an interface model aligned with the enterprise architecture are susceptible to hindering future integration efforts. Similarly, an interface model that does not address all data exchanges creates ‘islands of integration’ where only a subset of systems is integrated. Interfaces that involve code ‘owned’ by a division of an enterprise promote turf-holding and risk missed opportunities for integration.

## B. Interface Concepts

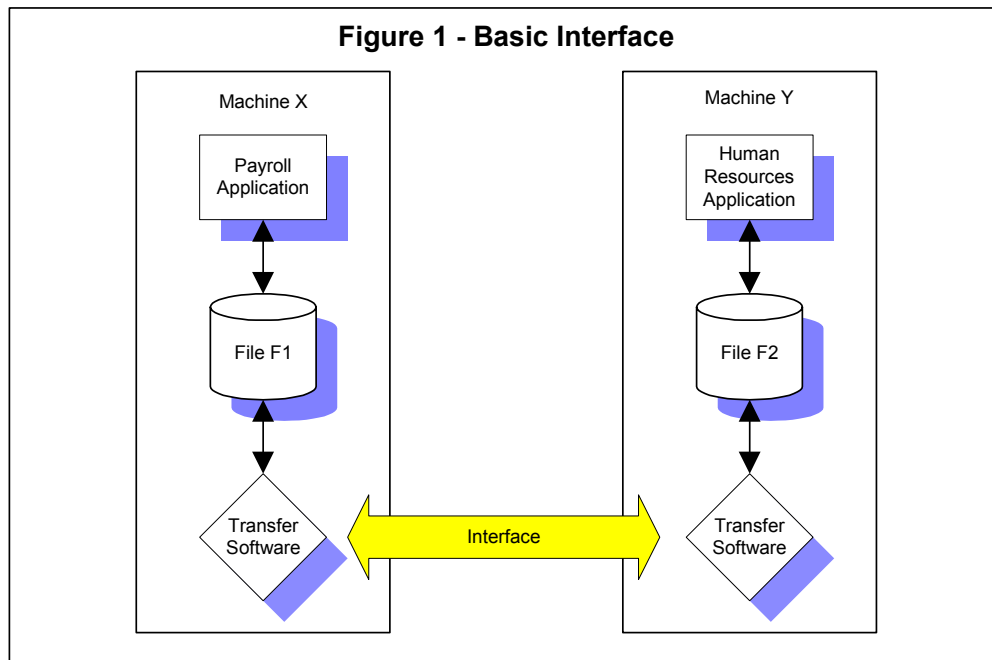
### Interface Technology Fundamentals

At the most abstract level, interfaces are used to move information from one information system to another so that the receiving system can use the information effectively. The core characteristic of all interfaces is the transmission of data elements, which at the lowest level can include just binary bits (1s and 0s) or at the higher level human readable information such as numbers and text. See **Figure 1** for a basic view of an interface scenario where a Payroll application and a Human Resources application send data to each other. Assume the payroll application stores its data in a local file F1. Now assume that the newer Human Resources application obtains its data by reading data in a file F2 whose format is different from that of F1. Further, the two applications are run on two different machines. Thus, a communication link between these two applications implies the use of transfer software and the reformatting of data between F1 and F2. This document uses the term ‘transfer software’ as the software that performs the interface transmission, transformation, and work flow; however, note that **Figure 1** generalizes the implementation of ‘transfer software’ because not all transfer software connects directly to the database.

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---



In an ideal interface, the systems involved speak the same language and do not require any translation of the transmitted data's syntax or semantics. The syntax standards define the format of messages (e.g. XML) and the semantic standards define the meaning of the data (e.g. ebXML). Adopting 'open' standards provides the use of widely accepted and supported formats, avoiding the need for translations. Open standards support interoperability, portability, and scalability and are not proprietary. However, more frequently there is a need to translate the data from the originating system to a format that is compatible with the receiving system. The format consists of the set of symbols and structures that define the physical implementation of data conforming to specified semantics and syntax. The data interchange mechanism is a set of tools, utilities, or application programming interfaces which perform the conversion into and/or out of a data interchange format.

The interface topic is pervasive throughout the Office of Management and Budget's FEA. The Service Interface and Integration service area of the FEA Technical Reference Model provides a foundation to describe the standards, specifications, and technologies for how agencies will interface (both internally and externally) with a service component.

Although there is a variety of interface models in use, there are core characteristics that must be in place for each of the interface models to be effective. For a list of these characteristics see **Figure 2**. For descriptions of the characteristics refer to the Glossary of Terms in **Appendix B**.

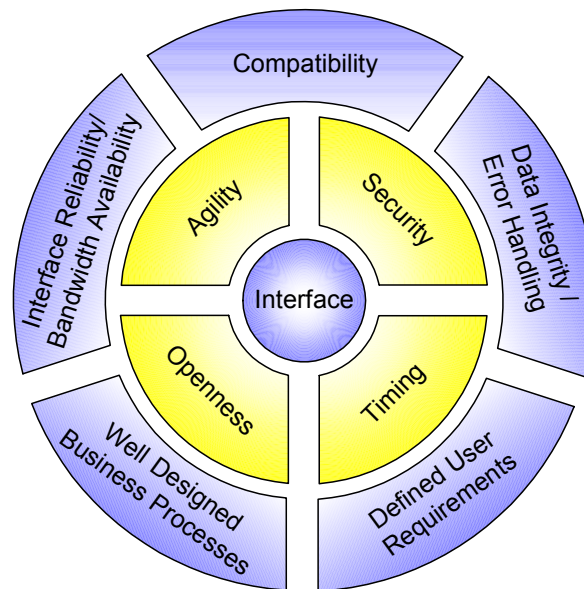


# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

Figure 2 - Characteristics of a Successful Interface



In the current technology environment it is no longer possible to talk about system interfaces without also discussing integration and interoperability. The Joint Financial Management Improvement Program recently held a forum to discuss "Keys to Successful Integration/ Interoperability of Business Management Systems". Although the forum focused solely on the government sector, there are two issues that are particularly applicable across all industries:

1. Funding mechanisms are a concern. Achieving interoperability of business processes is a complex undertaking, and not well supported by current government funding and accountability structures.
2. Communication remains a challenge. There is a lack of precision in the process. Vendor representatives stress that interoperability is a spiral rather than a linear development – systems must be defined to deal with ambiguity and change because it is impossible to identify all requirements and circumstances up front. Interoperability is an iterative process; assumptions cannot be cast in stone and changes create interdependencies which must be communicated to all stakeholders.[9] Selecting an interface model that adopts commercial standards helps reduce the impact of changes.

### Standards

Interface standards have been around since before computers were invented. In fact, one can look at the way electricity is provided in our homes to see an early interface standard. Electrical outlets are made with a standard configuration to provide interoperability with equipment. As with most interface standards, there is a limit to the deployment of the electrical outlet standard. For instance, when traveling overseas, a different electrical standard applies.

To see the importance of interface standards one can look to the Baltimore fire of 1904. Here the entire city almost burned to the ground because neighboring firefighters' hoses could not connect to fire hydrants as a result of non-standard coupling sizes and threads. Firefighters were called in from other cities, but when they arrived their hoses could not fit Baltimore hydrants. There was no standard thread size at the time for coupling hoses to hydrants. The fire destroyed 1,500 buildings over a 70-

## **Interfaces for Enterprise Solutions**

### **Army Enterprise Integration Oversight Office (AEIOO)**

---

block area. Following the fire, the National Bureau of Standards (now NIST) performed an investigation and identified 600 different coupling variations used in the country. Shortly thereafter a standard coupling was developed and deployed.[10]

Although we know standards help ensure that an interface will enable the connection of two systems, we do not know what standard an interface should use because the problem domain has become so vast. Interface standards run the gamut of formal to de jure to de facto. Formal interface standards are developed by accredited standards organizations such as the International Organization for Standards (ISO) and the American National Standards Institute (ANSI) or professional societies such as the Institute of Electrical and Electronic Engineers (IEEE). De jure interface standards are developed by legal authorities such as the National Institute of Standards and Technology (NIST) of the U.S. government. De facto standards come into being simply through widespread use and popularity. Other standards are developed by specific industries, such as the defense, health-care, banking, and automotive industries. For instance, the U.S. military developed MILS (Military Standards), which help to ensure that internal and external exchanges of data conform to standardized syntax and semantics to avoid errors in communication. MILS has been in use for nearly four decades, dwarfing any current interface technology standard, and has provided significant value over the years. However, it has grown outdated, partly due to its fixed length requirement (80 characters) and its lack of compatibility with commercial standards. Although the MILS standard needs to be phased out, it should serve as an example that standards are an essential component of exchanging data between disparate systems.

The benefits normally attributed to using standards are interchangeability, interoperability, portability, reduced cost and risk, and increased lifetime.[7] One interface standard currently gaining popularity is XML, which provides a way of describing data so it can be easily interfaced between systems. XML is an open standard developed by the World Wide Web Consortium (W3C). XML has many advantages, for instance, it can be read and understood by both humans and machines because it is stored in Unicode (text), it is platform independent and it is self-describing through the use of tags (i.e., labels, which is a key component of the DoD Net-Centric Data Strategy). Despite the recent popularity of Internet-based standards, EDI standards such as X.12 and EDIFACT continue to be the dominant interfacing standards used for electronic commerce (i.e., external transactions only), accounting for 95% of the world's transactions.[11] For example, DISA's DoD Electronic Business Exchange (DEBX) is an EDI hub for defense-related transactions that processes 30 million transactions per month.[12]

To date, standards have focused on protocols, such as how objects communicate; however, standards have not focused on business processes, or how objects perform. Efforts to develop business process standards must be strengthened to reap the full value of interface capabilities.

### **Interface Documentation**

Although it is widely known that documenting interfaces is critical to help ensure efficiency and effectiveness, interface documentation is rarely done sufficiently to accomplish that objective. Organizations usually have system environments that consist of heterogeneous components designed by different teams, each with distinct objectives. Interface documentation is necessary to help the components work together and support each other. Interfaces are the glue that holds the components together. Interdependencies also become evident through the process of creating and maintaining interface documentation.

While there are various opinions on the level of detail that should be documented for interfaces, there is no disagreement that core input/output information should be documented. Belliappa [13] provides a view on interface documentation for a system's external interface, stating that interface documentation should first detail a "big picture" view. Then the documentation should be the one point of reference that allows integrators to obtain any information about other components that might be

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

relevant to each other. This includes interface name and functions, supporting programs (i.e. methods/procedures), inputs, outputs, range of values for inputs and outputs where appropriate, exceptions that might occur, and exception handling. The interface documentation does not include detailed internal information of the components that might not be relevant to other components.

In the Defense arena, a format for interface documentation is prescribed in the DoD Architecture Framework (DoDAF). Within the framework's Systems View, the Systems Interface Description document (SV-1) is prepared for interfaces. The SV-1 first addresses the system level information and then serves to support further coordination with the Operational and Technical Views of DoDAF. DoDAF captures the far reaching implications of interfaces through this coordination at multiple 'view' levels. DoD also maintains a repository of information about implemented interfaces in the RICE repository to support reuse. RICE objects refer to data elements that may include object description and attributes, reference documentation, point of contact information, and development status. The purpose of the RICE repository is to create a vehicle in which COTS (commercial-off-the-shelf) implementations can leverage work already done, eliminate redundancy, and reduce efforts and costs.

### C. Interface Models

There are a few interface models that have been in existence for a number of years (e.g., point-to-point); however, the growth of the Internet has brought about advanced interface models to enable the sharing of information between systems. Many of the new, advanced interface implementations incorporate a mixture of other interface models, such as the Enterprise Service Bus incorporating Web Services. Listed below is a brief description of the widely used or emerging interface models.

- **Point-to-Point (P-2-P):** Model where one application exchanges information directly with *one* other application through the use of a tightly-coupled transfer program.
- **Hub and Spoke:** Model where a centralized software component (e.g. integration broker) routes messages and coordinates activities between connected applications. EAI products generally implement some form of this model.
- **Web Services:** Model that leverages open standards (e.g. XML) to enable organizations to provide and use 'services' via the Internet, both internally and externally.
- **The Accidental Architecture:** Model that results from accumulating a mixture of incompatible interface models, creating stove-pipes of information sharing.
- **Enterprise Service Bus (ESB):** Model that incorporates a variety of compatible standards-based interface technologies, including Web services, message oriented middleware (see below), transformation, and routing intelligence in a service oriented architecture.
- **Extract, Transform, Load (ETL):** Model supporting data integration (instead of application integration) through the extraction of data from source systems and storing it in a data warehouse to be used by other systems.
- **Message Oriented Middleware (MOM):** Model that uses a centralized software component to route asynchronous messages between disparate applications, largely reliant on message queues to provide temporary storage when the destination application is busy or not connected.


























































For a more detailed discussion of the interface models and their cost implications, refer to **Appendix A. Table 1** compares the key characteristics of the interface models listed above. The table shows that Web Services and Enterprise Service Bus are the most promising interface models, despite their newness. Only in very specific environments do Hub and Spoke, Extract, Transform, Load, and Message Oriented Middleware provide a case for their use. In almost no environments would an architect be able to justify the use of the Point-to-Point model or the Accidental Architecture. Despite

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

the computer industry's use of interfaces for many years, simplicity and low-cost are characteristics that continue to evade most interface models.

**Table 1 – Interface Model Characteristics**

	P-2-P	Hub and Spoke	Web Services	Accidental Arch.	ESB	ETL	MOM
 - Positive							
 - Negative							
<b>Architecture</b>	1 to 1 Distributed	1 to Many Centralized	1 to Many Distrib.	Mixture	1 to Many Distrib.	1 to Many Central.	1 to Many Central.
<b>Future Potential</b>							Mod.
<b>Maturity</b>		Mod.					
<b>Install Cost</b>					Mod.		
<b>Long-term Cost</b>							
<b>Agility</b>							
<b>Open Standards</b>							
<b>Vendor Indep.</b>							
<b>Loose Coupling</b>							
<b>Scalability</b>							
<b>Simplicity</b>					Mod.		Mod.

### D. Interface Cost Analysis

When Aunt Bea says she wants to buy a new Cadillac because she saw an ad with one listed for only \$30,000, is it really going to cost just \$30,000? The likely answer is no, due to additional purchase costs such as tax, delivery, and equipment options. These costs could quickly add up to \$7,500, 25% more than what Aunt Bea anticipated. Then what about her long-term costs? Does her chosen Cadillac have poor reliability ratings and require expensive maintenance? Will her navigation system require her to purchase and install an expensive update every time there is a new release? Is her Cadillac the 'hot item' now but likely worthless in a couple years because it's so outdated?

Many of these same questions could be asked when preparing a cost estimate for a new interface. Chief Information Officers may offer a reasonable estimate of *one million dollars* for the cost of a complex interface in a large scale enterprise. However, to determine a more precise estimate of the cost of an interface, one must look under the hood to analyze the details of implementing the interface. There are many components that can significantly affect the cost of an interface, just as there are many costs that can affect the price of a car. A cost estimate should consider costs for requirements analysis, feasibility study, design, build, configuration, test, and deployment. Possibly even more important are the long-term costs of an interface. These may be more significant than the

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

---

implementation costs due to direct costs (e.g., maintenance) or indirect costs (e.g., inability to further integrate the enterprise).

One of the most common mistakes organizations make when interfacing systems is assuming that operating environments will not change dramatically or that their interface model can handle any change that may occur. If the integration is not effective at accommodating change at all levels, integration explosion syndrome develops, wherein the organization is widely-dependent on the integration but cannot increase functionality without devoting substantial resources to development.

#### Economic Cost Estimating Methods

Exhaustive research has been performed on methods to estimate the cost of developing software systems; however, there have not been significant studies on methods to estimate the cost of interfaces. Further contributing to the ambiguity of interface costs is the number of new interface technologies being deployed as a result of developments such as the Internet and object-oriented programming. Current interfaces are built in a significantly different manner than interfaces in the past. Historically, point-to-point interface programs requiring significant coding were the norm; however, current interface technology allows the use of software tools, largely eliminating the need for costly coding.

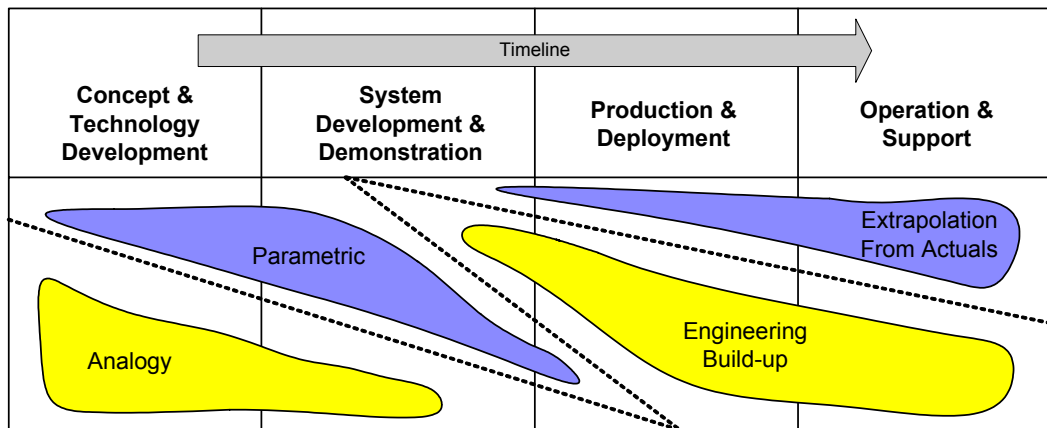
Although traditional software cost estimating methods can still be used to estimate costs for interfaces, emphasis should be placed on tailoring the estimating methods using the interface cost drivers below. Common software estimating methods include function point analysis, analogy, parametric (e.g., cost per line of code), and build-up (i.e., aggregating lower-level estimates).[14] A hybrid of the approaches may be the best method because each interface environment is so distinct. One method may work well in a certain situation, but that same method may be a poor method for another interface. Another important point is that each of the different interface models has specific cost factors that heavily influence the cost of that type of interface. For instance, Web services may not require significant custom coding because of supporting tools; however, that does not mean it will be a more cost effective interface model if the organization does not have staff with Web services experience.

Each of the costing methods has optimal times for use in the interface lifecycle (refer to **Figure 3**). During the concept and technology development phase, the **analogy** method is most appropriate due to a lack of concrete information (other than past implementations that can be compared). During the system development and demonstration phase the **parametric** method is optimal because cost drivers are now more defined and can be used for statistical inferences using data from multiple implementations. During the production and deployment phase, the **engineering build-up** method is ideal because more low-level detailed information about the implementation is available and can be rolled up to produce higher-level estimates. Finally, during the operations and support phase, the **extrapolation from actuals** method is ideal because actual cost information is accessible for similar activities on the same project.[14] There is some overlap in phases for using the four methods, as shown in **Figure 3**.

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

Figure 3 - Lifecycle of Costing Methods



For each type of interface evaluated, system integrators (or other staff familiar with the cost implications of the various interface technologies) should identify the set of cost drivers that will have the most significant influence on the cost of a particular interface. These cost drivers can be categorized as functional, technical, or organizational, and include the following:

### Cost Drivers

#### Functional

- Degree of integration that will be performed (e.g., amount of business process logic involved)
- Function points (from Function Point Analysis) (refer to the Glossary of Terms in **Appendix B**)
- Volume of data exchanged between systems (number of messages/transactions)

#### Technical

- Number of lines of source code (SLOC) (count does not include comments, blanks, and continuation lines) Note: many modern interface technologies do not require code development.
- Complexity of the data structure (e.g., relational, flat file, EDI)
- Number of systems affecting the interface
- Number of interfaces developed similarly
- Maturity of the technology being used
- Technical complexity, including:
  - Complex logic and/or calculations
  - Multiple input parameters
  - Multiple transaction formats
  - Complex transformation requirements
  - In-depth programmer testing

#### Organizational

- Amount of budget available
- Time required for development (development hours)
- Number of physical locations and/or departments involved
- Capability/experience of functional and technical team
- Labor rates

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

---

Each of the costing methods is susceptible to the same vulnerability, which is placing too much reliance on the model. The models make it easy to guess at cost drivers, but it is important to have the best understanding possible of the cost drivers in order to get the most accurate estimate.[15]

As interfaces approach the operation and support phase (i.e., the last phase), it is critical that costs continue to be tracked and reported to support future cost estimates and comparisons. Identifying the cost of internally developed interfaces is a basic calculation using the budget and the completed project plan. The costs are usually calculated from labor hours for analysis, design, coding, testing, and integration while the fixed costs are used for hardware, installation, and training. However, when an interface involves 'purchased' interface technology, such as vendor provided interface adapters, the coding effort is much less costly. Although organizations account for this change in cost estimates, they often do not properly account for the increase in costs to internally integrate the purchased software. It is widely assumed that as soon as a new technology is put into place it starts paying dividends right away. However, experienced system integrators know post go-live support can be extensive before the full benefits of a new interface technology are realized.

Definitive interface costs, especially internal costs, are elusive because there is such an overlap with many other cost areas involved in maintaining a connected enterprise. This was painfully obvious while performing research for this document, as most of the cost analysts interviewed were not able to identify interface costs for the large system implementations they administered. Although there are many variables that can affect an estimate, it is worth discussing sample interface costs to provide a point of reference. **Table 2** provides interface cost estimates from an established DoD-wide Blanket Purchase Agreement, for fiscal year 2004, with a contractor for an SAP ERP implementation.[16] When analyzing the estimates above, it is essential to keep in mind their context. They are *estimates* specifically prepared for interfaces between an ERP and an outlying system. As described above, there are many other types of interfaces and they can have significantly higher cost estimates, such as pure-play EAI or point-to-point.

**Table 2 – Interface Cost Estimate  
(between ERP & outlying system)**

Task	Acquire-to-Retire Business Process	Procure-to-Pay Business Process
Interface Analysis	No line-item avail.	No line-item avail.
Interface Design	No line-item avail.	No line-item avail.
Build Interface Programs (all required)	\$259,748	\$1,069,779
Interface Unit Test	<i>Per Interface:</i> \$23,858 (simple) \$48,018 (medium) \$93,016 (difficult)	<i>Per Interface:</i> \$23,858 (simple) \$48,018 (medium) \$93,016 (difficult)
Interface Maintenance	Not included	Not included

Note: In the table above, simple, medium, and difficult complexity are based on the number of end-to-end processes, ranging from 1-15, and the number of resources, ranging from 1-30.

Instead of estimating interface costs per implementation task, another contractor chose to estimate total interface costs for each individual interface in their Blanket Purchase Agreement.[17] These estimates are based on economies of scale where many similar interfaces are created by the same team. Again, note that these are estimates for an interface between SAP and an outlying system

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

---

where much of the interface development may have already been created and made available by the ERP vendor. This contractor estimated a total interface cost for each of the three levels of interface complexity as follows:

Low Complexity	\$ 3,456 per individual interface
Medium Complexity	\$ 10,370 per individual interface
High Complexity	\$ 20,741 per individual interface

### Counting Interfaces

There is a general misconception regarding the identification of interfaces. It is not uncommon for an interface between two systems to be referred to as the 'X' interface, when really there might be an 'X', 'Y', and 'Z' interface between those two systems. For instance, an interface between a U.S. Army logistics system and a back-office accounting system may involve one interface with a real-time transmission of munitions orders and another interface for an overnight batch transmission of accounting journal entries. While the two transmissions of data are between the same two systems, they are actually different interfaces and should be managed distinctly. The characteristics listed in **Table 3** help distinguish one interface from another (note: these are documented in the DoDAF SV-6 Systems Data Exchange Matrix).

**Table 3 – Interface Characteristics**

Characteristics	Examples
1. Name	Equipment Maintenance Overages
2. Purpose	Post daily overages for maintenance costs
3. Systems Involved	'ABC' system and 'DEF' system
4. Transmission Mechanism	Manual, FTP, Messaging, Web services
5. Data Volume	# of records, # of transactions
6. Quality of source data (i.e., errors)	Good, average, poor
7. Complexity of data	Simple, average, complex, very complex
8. Type of Interface	Real-time, pseudo-real-time, batch
9. Direction of Interface	Inbound, outbound, bi-directional
10. Frequency	Daily, weekly, monthly

### E. The Road Ahead

A prominent (and positive) trend in the future of interfaces will be the continued adoption of standards and open protocols. Commercial standards, such as XML and SOAP, appear to be the wave of the future and industries such as health-care and military will be forced to follow along.[18] This adoption of standards allows applications to interface with each other without knowing the details of each other, following the concept of loose-coupling. In the past, System A had to know about System B in order to interface with it. However, the use of standards allows System A to interface with System B without knowing the details of it.[19] Loosely coupled interfaces provide the flexibility needed in today's constantly changing information technology environment.

Another trend in the future, somewhat by default and somewhat by design, will be the availability and use of a mix of interface models in the marketplace.[20] Due to an organization's specialized needs when interfacing systems, there will never be an interface model with complete market dominance.



## **Interfaces for Enterprise Solutions**

### **Army Enterprise Integration Oversight Office (AEIOO)**

---

For instance, there will never be a Microsoft Windows of the interface model world. However, even in the Microsoft Windows example, there are divergent operating systems such as Mac OS and Linux.

Software vendors are constantly attempting to develop new techniques to efficiently and effectively create interfaces. One technique that is gaining traction is the meta-data approach. The meta-data approach simplifies the development process by allowing system integrators to input information about business processes into a graphical user interface (GUI) tool, which then develops the adapters to which applications interface. The GUI allows data relationships to be defined at the logical/meta level instead of the data level. Thus, there is no source code to program and maintain. This approach makes the interface development process easier to understand, easier to manage, easier to customize, and easier to evolve. Products such as Oracle's InterConnect tool and Evolutionary Technologies International's (ETI) Solution v5 use the metadata/no code approach. DoD's Standard Procurement System (SPS), an automated contracting system for \$48 billion in goods and services, recently had twenty-two interfaces that were created using ETI's Solution tool.

When asked whether future IT environments will incorporate more or fewer interfaces, the natural response would be to say that the number of interfaces will grow as information services expose themselves through well described services such as Web services. However, many organizations will continue to migrate towards the ERP approach, bringing applications under the same system and eliminating interfaces. This follows the strict definition of 'integration' in the ERP context, where interfaces are eliminated because the interfacing of business processes occurs within the ERP software. This concept of integration provides immediate and long-term cost savings. On the other hand, some organizations will continue to follow the 'best of breed' approach and use individual applications that are optimal for their needs. These organizations will be forced to continue, and possibly increase, the use of interfaces. However, the antiquated interfaces, namely point-to-point interfaces, will be minimized and replaced with more current technology to reduce sustainment costs.[21]

As organizations attempt to avoid the pitfalls of the 'accidental architecture', they will realize that interface model decisions should be defined prior to the approval of new system acquisitions. The practice of assessing interface model and standards compliance during the funding process will continue to be adopted as integration amongst systems continues to gain significance. DoD is taking steps to block system initiatives (although not specific to interfaces) estimated to cost \$1 million or greater, that are not aligned with DoD's Business Enterprise Architecture (BEA). The Defense Appropriations Act of 2004 requires a BEA compliance assessment for all system investments of \$1 million or greater.[22] This practice promotes more of a lifecycle approach where interface planning is performed before any money is spent on implementing a new system, instead of after systems are implemented and exponentially more money must be spent to integrate systems. This full lifecycle approach is one of the primary tenets of portfolio management, which is an activity mandated by the Deputy Secretary of Defense, as of March 22, 2004.[23]

## **F. Wrap-up**

Depending on with whom you speak and what terminology you use, interfaces may be frowned upon or considered essential. Older interface models, including point-to-point, are expensive to maintain and often do not take advantage of increased integration that can be gained from interfacing more than just flat file data (e.g. semantics). While newer interface models, such as EAI and Web services, are unproven on a grand scale, they promise to provide organizations better efficiency and effectiveness through adoption of standards and object-oriented technology. Interface technologies that adopt standards, such as XML, will gain the attention of Chief Information Officers for the coming years.

Newer interface technologies provide vastly improved capabilities, but their significant implementation costs are a deterrent for some organizations. However, with the implementation of an appropriately planned interface model, organizations will see that the long term value outweighs the upfront costs.

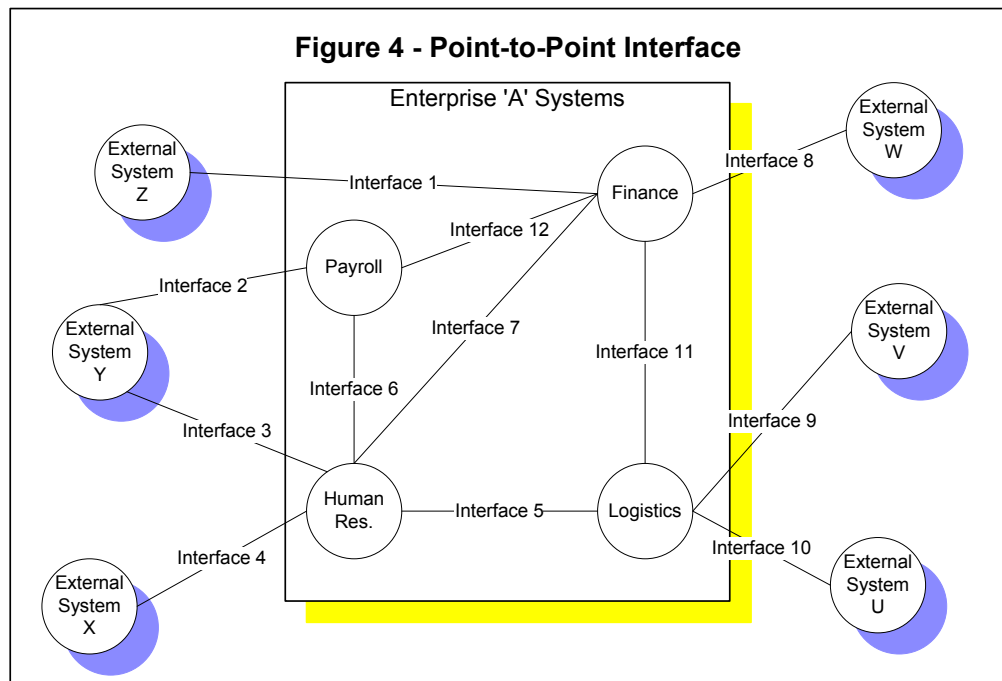
# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

### Appendix A – Interface Model Descriptions

#### Point-to-Point

System interfaces originated with the point-to-point model where one system exchanges information directly with *one* other system. Depending on the disparity between the two systems interfacing, there is some degree of data translation or transformation that must take place for the two systems to communicate. While translation involves simply changing the format of a data element, transformation involves mapping a data element from source system to destination system (e.g. purchase order address to shipping address). Additionally there must be software to transfer the data between the two systems. This brings us to the primary drawback of the tightly coupled nature of the point-to-point model. The transmission and translation is performed by a software program, and generally must be tailored for each specific interface. This short-sighted approach lacks compatibility and agility, and also provides little opportunity for reuse. As the number of interfaces grows, the amount of effort to develop and maintain these interfaces does as well. Because systems generally continue to be added, the information systems of many organizations resemble what is called *spaghetti systems*.<sup>[24]</sup> Refer to **Figure 4** for an example of how point-to-point models lead to spaghetti systems.



Batch interfaces discussed earlier are also classified as point-to-point because they still consist of an interface from one system to *one* other system. Although the point-to-point model has many drawbacks, it can be useful in organizations that are very small, non-complex, have very low technology funding, and have relatively static system needs.

#### *Cost Implications:*

Point-to-point is a high-cost, high-risk interface model that requires dramatically more resources to maintain than a more unified approach. While the point-to-point approach may provide the 'quick-fix' solution in very small environments, it becomes expensive to scale to multiple systems. The point-to-point approach requires significant operations and maintenance costs because of the numerous

# Interfaces for Enterprise Solutions

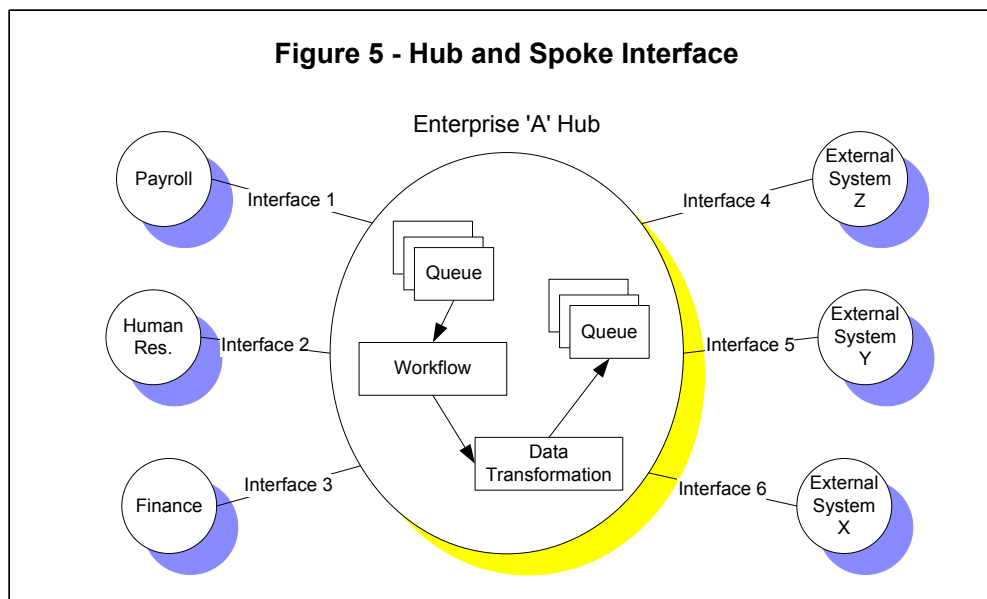
## Army Enterprise Integration Oversight Office (AEIOO)

---

transfer and translator programs required and the complexity of adding new systems. Because point-to-point provides few opportunities for reuse, it suffers from poor economies of scale.

### Hub and Spoke

The advancement of system interfaces/integration is no longer limited to the system level, but also includes the semantic level. Therefore, the nature of what is exchanged has changed from simple character strings to business elements (e.g. invoice, purchase order). This new meaning delivered with information requires complex systems and places a heavier load on systems. A solution to handle this heavier load on systems is to use a centralized model, analogous to a bicycle wheel's hub and spokes. The hub represents the center of interface activity and the spokes represent the links that connect the hub to various internal and external systems. Refer to **Figure 5** for an example of the hub and spoke model. This model provides a one-to-many communication ability as opposed to the one-to-one ability of the point-to-point model.



The hub provides the following critical functions:

- transforming data from the sender's format to the receiver's format,
- handling message queues *from* senders and *to* receivers,
- providing workflow directing messages to the intended receiver, via the queue, and
- enforcing security and access rights to information (LDAP can be used here).

The spoke constitutes the link between the hub and the applications. The spoke handles the protocol for circulating messages and provides the adapter software to connect systems.

EAI solutions, which began to appear around 1997, typically follow a form of the hub and spoke approach. This approach relies heavily on middleware software to exchange messages (consisting of data, etc.) between systems. Middleware consists of software agents acting as an intermediary between different application components. EAI is more ideal for single organization implementations because of its centralized control approach and its tendency to suffer organizational boundary problems, which are sometimes caused by limitations in spanning firewalls and network domains. EAI has also suffered from steep learning curves and expensive barriers to entry on individual projects.[25] In recent years, EAI initiatives have had moderate success with limited future potential. Vendors such as IBM, SeeBeyond, webMethods, and Microsoft are heavily promoting their EAI solutions to integrate

## **Interfaces for Enterprise Solutions**

### **Army Enterprise Integration Oversight Office (AEIOO)**

---

disparate systems. Due to the proprietary nature of the vendor provided code, EAI generally lacks the openness provided by Web services discussed below.

Tibco is another leading EAI vendor, but instead of using the hub and spoke layout, Tibco uses an information bus layout. An information bus is analogous to a large pipeline connecting an organization's systems and is beneficial because it eliminates the hub as a bottleneck. As displayed by Tibco, not all EAI solutions follow a hub and spoke approach. In fact, many EAI solutions are a hybrid of the hub and spoke approach, Web services (discussed below), and other interface models. For a review of EAI products prepared by the Army Enterprise Integration Oversight Office (AEIOO) refer to [http://www.army.mil/aeioo/docs/AEIOO\\_EAI\\_POV.pdf](http://www.army.mil/aeioo/docs/AEIOO_EAI_POV.pdf).

In addition to pure-play EAI vendors whose specialty is EAI software, ERP vendors have also developed add-on products to their ERP software that aim to integrate all of an enterprise's business processes, not just those encompassed by the ERP package. For instance, SAP's NetWeaver and Oracle's InterConnect are integration and application platforms that use the hub and spoke model with an integration broker as the intermediary between the ERP and outlying systems. The advantage of using these products is that they are integrated 'out-of-the-box' with the ERP software. However, for the outlying systems adapters must be developed to interface with the hub. If the outlying systems are XML compliant, interoperability is possible since most products are XML based.

#### *Cost Implications:*

Organizations looking to implement a hub and spoke solution are faced with a hefty initial price. The cost for software licensing can range from \$250,000 to well over \$1 million. This typically carries with it heavy implementation costs, often five to twelve times the cost of the software licenses.[25] Hub and spoke product vendors, like other advanced interface technology vendors, state that it is a strategic investment because it lets companies leverage previous technology investments, add new technologies at a lower cost, and integrate the disparate systems. Despite that consideration, EAI's proprietary nature and lack of a standardized platform providing general-purpose use across an extended enterprise make them undesirable for widespread use in large organizations. It may also be unfavorable to medium size organizations because it is too expensive.

When comparing pure-play EAI products versus ERP EAI products, costs are highly impacted by the specific system environment. For instance, if an environment consists of applications that can natively communicate with the ERP EAI product, significant savings can be gained by using the ERP vendor's EAI product because of the 'out-of-the-box' integration. ERP EAI products are becoming very attractive to system architects as the vendors continue to adopt open standards paralleled with the 'out-of-the-box' ERP integration.

#### **Web Services**

The proliferation of the Internet has created a vast network that can be used for interfacing information and services. Web services is an interface model that attempts to maximize the value of the Internet by adopting standard protocols to achieve compatibility between applications using disparate platforms and environments. OASIS (Organization for the Advancement of Structured Information Standards) and the W3C are the steering committees responsible for the architecture and standardization of Web services. Some technologists do not actually consider Web services to be an interface model since its forte is its ability to expose functionality on the Internet; however, it should be considered an interface model since it provides a standardized way of integrating Web-based applications (i.e., program-to-program communications).

In a nutshell, Web services enable organizations to provide and use 'services' via the Internet, both internally and externally. A 'service' can be defined as a package of closely related standardized functions, which are called (i.e. a program is 'called') repeatedly in a similar fashion, and should therefore be implemented by a dedicated facility, which can be specialized to perform them.[26] The

## Interfaces for Enterprise Solutions

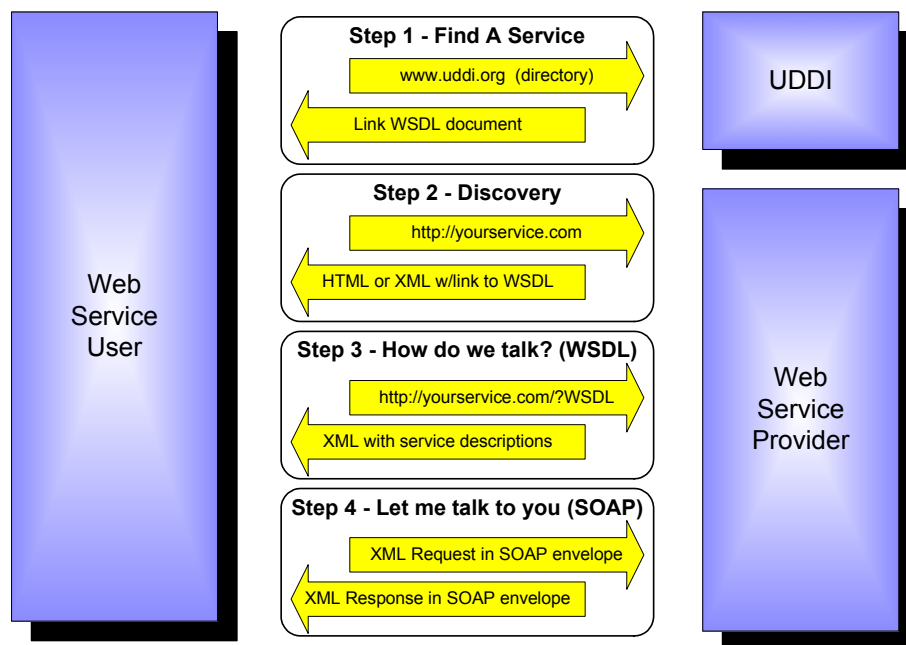
### Army Enterprise Integration Oversight Office (AEIOO)

---

dedicated facility receives the request, processes it, and returns a response. An often-cited example of a Web service is that of a stock quote service, in which the request asks for the current price of a specified stock and the response returns the stock price. In external Web services the 'call' and 'service' originate in different organizations. Conversely, large organizations may use Web services internally where service providers and consumers are internal to the organization.

Web services attains its 'openness' through the use of XML (eXtensible Markup Language) and one or more of the following three protocols to interface with other parties over the Internet: UDDI, WSDL, and SOAP. XML, as discussed above, is a specification for capturing and labeling information exchanged. UDDI (Universal Description, Discovery, and Integration) is the standard for publishing and discovering Web services (similar to the Yellow Pages). For the U.S. Army, this UDDI directory could reside in AKO (Army Knowledge Online), as well as the actual Web services (i.e., host applications). WSDL (Web Service Description Language) is the language that describes the location and interfaces that a particular service supports. SOAP (Simple Object Access Protocol) provides the explicit serialization protocol used in service exchanges. Refer to **Figure 6** for a diagram of the process flow of these protocols. However, Web services do not currently address the transfer of meaning or intent and, therefore, it requires parties to agree on semantic standards. Also, security standards for Web services are currently evolving to provide a unified mechanism for authentication. The WS-Security 1.0 standard was released in 2004 to provide specifications for authentication and confidentiality.

**Figure 6 - Web Services Process Flow**

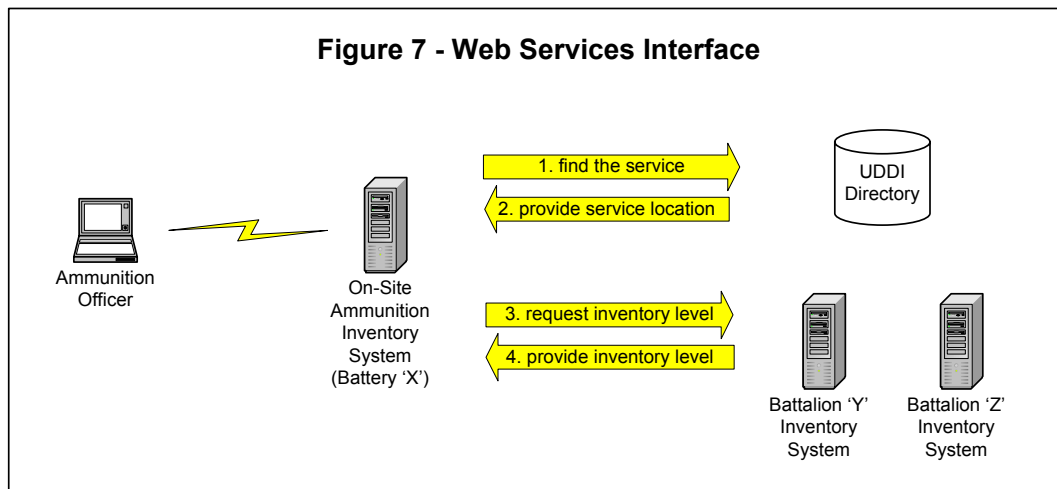


Web Services is key to DoD's Net-Centric Data Strategy to support its goal of increasing combat power of military force by creating a network that connects sensors, decision support systems and weapons systems with the people that use them.[27] The attainment of net-centricity involves the fundamental shift to a distributed, Web-enabled, service-oriented paradigm, requiring the support of a ubiquitous network environment. The primary benefit of Web services is that it provides efficient access to globally distributed services, and potentially eliminating redundant services. It's like cleaning out your desk and realizing you have four staplers – then selecting one to keep.

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

**Figure 7** shows a military application of Web services. Note: this is a simplified example and does not reflect actual systems in place. Web services could support an Ammunition Officer restocking munitions in preparation for a mission. Instead of implementing a point-to-point interface between all of the nearby artillery battery inventory systems, Web services would simplify the activity through its standard interface. The on-site ammunition inventory system would contact the UDDI directory to find the location of available battalion inventory systems. It would then 'call' the services to request the inventory level. Once munitions are located, the on-site inventory system could be prompted to locate the nearest source of transportation via Web services provided by transportation systems.



Microsoft is a leading participant in the Web services market through their .Net platform. Many of the other large software companies compete in the Web services market, including IBM, SAP, Oracle, BEA, and Sun Microsystems.

#### *Cost Implications:*

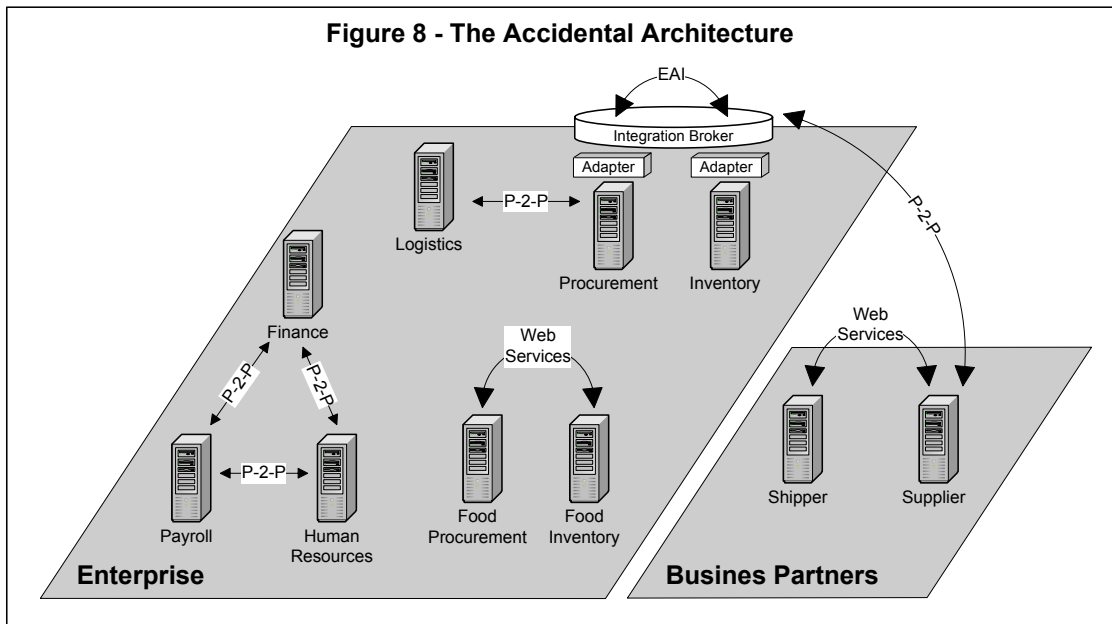
Web services provides cheaper integration than point-to-point through its openness, compatibility, and agility characteristics. As opposed to point-to-point implementations, Web services do not require code development and maintenance for transformation and adapter programs between each pairing of systems. Many argue that Web services is more economical than the hub and spoke approach due to its low start-up costs, but this can greatly vary from implementation to implementation. Further, since Web services is an industry supported movement, contrary to the individual vendor support for hub and spoke products, its stability has an intangible impact on long-term costs. Also, Web services consists of relatively simple technology so there is a short learning curve.

#### **The Accidental Architecture**

The accidental architecture is something that nobody sets out to create, but almost everybody has. It is the result of years of accumulating one-of-a-kind interface solutions.[25] Applications are locked into an inflexible interface model, creating stove-pipes of information sharing, otherwise known as 'islands of integration', because the collection of interface technologies deployed cannot provide the complete solution. Many of these environments start out with a deliberate interface model (typically point-to-point), and then over time additional interface technologies are added. This often occurs in system environments maintained by large numbers of people with multiple departments responsible for information technology. Refer to **Figure 8** to see how the accidental architecture incorporates a variety of interface models, such as point-to-point, Web services, and EAI, creating islands of integration.

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)



Problems associated with the accidental architecture include unreliability, poor performance and scalability, weak security, troubleshooting difficulties, poor redundancy and resiliency, and lack of standard monitoring and management. The two most important steps to help avoid the accidental architecture are to adopt an appropriate interface model before implementing replacement enterprise systems and to avoid deviating from the adopted interface model. Far too often organizations implement enterprise systems and then attempt to integrate their system environment, instead of planning the integration first and then implementing an enterprise system.

#### *Cost Implications:*

Few will argue with the notion that the accidental architecture is the most costly approach. To start with, it results in lost business process integration opportunities because of conflicting interface models. Second, making changes to the accidental architecture can become increasingly challenging and costly over time as the variety of interface solutions increases. Third, the accidental architecture requires purchasing and developing software and maintaining staff to support multiple unnecessary interface technologies.

#### **Other Interface Models**

##### Enterprise Service Bus (ESB)

The Enterprise Service Bus (ESB) concept was released in 2002 and is touted as the next generation of integration middleware. It aims to provide the underpinnings for a loosely coupled, highly distributed integration network. ESB incorporates four main components: message-oriented-middleware, Web services, transformation, and routing intelligence. A defining concept in ESB (as opposed to Web services) is its ability to remove process routing logic from the applications and place it in the bus so the applications can focus on application logic. This allows intelligent routing via the addition of conditional decision points based on the inspection of XML content within a message. Another defining characteristic of ESB is that messages are generally sent asynchronously so the sending application can send a message and continue to its next task without waiting for a response (although there still requires a message acknowledgement).

An ESB enables the implementation of a Service Oriented Architecture (SOA), which is a set of loosely coupled services working together seamlessly over a network to provide functionality to end



## **Interfaces for Enterprise Solutions**

### **Army Enterprise Integration Oversight Office (AEIOO)**

---

users. Legacy applications do not have to be modified to connect to the bus as connectivity can be achieved through multiple protocols, client Application Programming Interface (API) technology, legacy messaging environments, and third-party application adapters.[25] Although ESB allows incremental adoption, timing becomes an issue for large decentralized enterprises such as DoD, or even just the U.S. Army, because each of the user groups must have the same requirements from a specified service and the service provider must meet those requirements all at the same point in time. Due to its recent inception, this model has not been widely deployed on large projects yet; however, ESB shows great potential for the future. While the concept of ESB was pioneered by Sonic Software, integration companies such as webMethods, SeeBeyond, and IBM are establishing their presence in the ESB space.

#### Extract, Transform, & Load (ETL)

ETL is focused on enterprise *data* integration, as opposed to enterprise *application* integration. ETL involves extracting data from source systems and storing it in a data warehouse to be used by other systems. Data moves from the source systems to the engine (data warehouse), where transformation and reorganization are done, then loaded in the target system. ETL products like those offered by Informatica, Ascential, and Business Objects facilitate the consolidation of data required for data warehouses. They assume the target system is a relational database management system (RDBMS). A major impact of the ETL model is that, depending on the vendor, it can have serious implications on scalability and performance since all data must pass through the RDBMS engine. ETL appears to have surpassed its peak in potential.

#### Message Oriented Middleware (MOM)

Message Oriented Middleware is a category of inter-application communication software that relies on asynchronous message passing (i.e., timing of request & reply are not linked). This differs from most interface models (except ESB) where a request is sent to a system prompting it to send a response back. Most MOM implementations are based around a message queue system, although there are implementations that rely on a broadcast (all recipients) or multicast (multiple selected recipients) messaging system. MOM software performs storage, routing, and transformation, all generally managed through coding at a low-level. A lack of MOM standards and expensive coding requirements contribute to MOM's lack of widespread deployment. Products that employ MOM technology include IBM Websphere MQ, Oracle Advanced Queuing, Tibco Rendezvous, Microsoft Message Queuing, BEA Tuxedo, and Sonic Software's SonicMQ.

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

### Appendix B – Glossary of Terms

**Agility.** Agility is the ability to adapt to different interface requirements, the ability to modernize interfaces at different paces (i.e., forward and backward compatibility), and the ability to bust apart interface components and realign.

**API.** An Application Programming Interface (API) is a set of definitions of the ways in which one piece of computer software communicates with another. It is a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software allowing developers to use programs already created for commonly used functions.[11]

**Business Logic.** It is the core of any application, providing the expression of business rules and procedures (e.g. the steps and rules that govern how a sales order is fulfilled). As such, business logic includes the control structure that specifies the flow for processing business events and user requests.[8]

**Compatibility.** A quality two systems have if they can communicate with each other using similar syntax and/or semantics.

**COTS.** The term Commercial-Off-The-Shelf software describes software or hardware products that are ready-made and available for sale to the general public. [11]

**Data Integrity.** A characteristic possessed by an interface when data is unchanged from its source and has not been accidentally or maliciously modified, altered, or destroyed.[11]

**Data Interchange Format.** An intermediate format required to convert data from a set of input formats which share common semantics and syntax to a set of output formats without loss or distortion of content.[28]

**Data Interchange Mechanism.** A set of tools, utilities, or application programming interfaces which perform the conversion into and/or out of a data interchange format.[28]

**Enterprise.** Organizational entity of any size (especially large organizations) that uses information networks to interact with employees, vendors or customers.

**Enterprise Integration.** See Integration.

**ERP.** Enterprise Resource Planning (ERP) systems are management information systems that integrate and automate many of the business processes associated with the operations or production aspects of a company. Also referred to as an enterprise system.

**Format.** The set of symbols and structures which define the physical implementation of data (e.g., Structured Query Language (SQL)) conforming to specified semantics and syntax.[28]

**Function Point Analysis (FPA).** FPA has been proven as a reliable method for measuring the size of computer software to assist in cost estimating.[29] Function Point Analysis is a structured technique of problem solving. It is a method to break systems into smaller components, so they can be better understood and analyzed. Function points are a unit of measure for software much like an hour is to measuring time or miles are to measuring distance.[30]

**Interchangeability.** Interchangeability is the ability to interchange components with different performance and cost characteristics. In this way creating multiple versions of a system in which one or more components are interchanged is possible because the adoption of these standards makes the interchange possible. An example is the personal computer.

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

**Integration.** (as in Enterprise Integration) Integration is the vertical and horizontal alignment of plans, business processes, and information systems across organizations and functional boundaries to provide a competitive advantage.[31]

**Interface.** An interface provides the capabilities of communicating, transporting and exchanging information through a common dialog or method.[32] A software interface is the languages and codes that the applications use to communicate with each other and with the hardware intended to eliminate the need for human intervention.

**Interoperability.** It is the ability of software and hardware on different machines from different vendors to share data. It allows a system to operate with a wider variety of external systems that have also adopted the same conventions/standards.[6]

**LDAP.** Lightweight Data Access Protocol is a standard protocol for describing access rights in networks.

**Middleware.** Middleware consists of software agents acting as an intermediary between different application components. It is used most often to support complex, distributed applications.[11]

**Net-Centricity.** It is the realization of a networked environment (including infrastructure, systems, processes, and people) that enables a completely different approach to warfighting and business operations. [33]

**OASIS.** The Organization for the Advancement of Structured Information Standards (OASIS) is a global consortium that drives the development of e-business and Web service standards.[11]

**Openness.** It is an attribute of a technology component that is based on standards. This allows disparate systems to communicate with each other easily.

**Portability.** Software systems adopt the standards necessary to run on multiple platforms with varying hardware or operating systems.[6] The Java language is a classic example of portability because its programs do not have to be run on any specific operating system.

**Portfolio Management.** The processes, practices and specific activities to perform continuous and consistent evaluation, prioritization, budgeting, and finally selection of investments that provide the greatest value and contribution to the strategic interest of the organization.[34]

**Semantics.** The content or meaning of data transmitted in an interface.

**Service Component.** Modularized service-based applications that package and process together service interfaces with associated business logic into a single cohesive conceptual module.[35]

**Service Oriented Architecture (SOA).** SOA is a set of loosely coupled services working together seamlessly over a network to provide functionality to end users.

**Syntax.** The format controlling the symbols and structures used to describe data.

**Timing.** A characteristic of real-time interfaces that maintain a transmission pattern in accordance with the designed sequence.

**Transformation.** Data transformation is the process of creating a correspondence between data elements of a source schema to (often different) data elements in a destination schema. An example

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

---

of data transformation is to map shipping and billing address information from a purchase order to an invoice. [36]

**Translation.** Data translation is the process of changing the format of a data element. [36]

**W3C.** The World Wide Web Consortium (W3C) is a consortium that produces standards—‘recommendations’, as they call them—for the World Wide Web. The Consortium is headed by Tim Berners-Lee, the original creator of URL (Uniform Resource Locator), HTTP (HyperText Transfer Protocol) and HTML (HyperText Markup Language), the principal technologies that form the basis of the Web.[11]

**Web Services.** The technologies involved in exchanging data between different applications and systems over an IP (Internet Protocol) network.

**XML.** eXtensible Markup Language. Specification developed by the World Wide Web Consortium (W3C) used to describe many different kinds of data. Its primary purpose is to facilitate the sharing of structured text and information across the Internet.

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

### Appendix C – Acronyms

API	Application Programming Interface (see <b>Appendix B - Glossary of Terms</b> )
BEA	Business Enterprise Architecture
CIO	Chief Information Officers
CORBA	Common Object Request Broker Architecture
COTS	Commercial-off-the-Shelf (see <b>Appendix B - Glossary of Terms</b> )
DISA	Defense Information Systems Agency
DoD	Department of Defense
DoDAF	Department of Defense Architecture Framework
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ETL	Extract, Transform, & Load
FEA	Federal Enterprise Architecture
FTP	File Transfer Protocol
GUI	Graphical User Interface
LDAP	Lightweight Data Access Protocol (see <b>Appendix B - Glossary of Terms</b> )
MOM	Message Oriented Middleware
OASIS	Organization for the Advancement of Structured Information Standards
NIST	National Institute of Standards and Technology
RDBMS	Relational Database Management System
RICE	Reports, Interfaces, Conversions, Extensions
SOA	Service Oriented Architecture
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

# Interfaces for Enterprise Solutions

## Army Enterprise Integration Oversight Office (AEIOO)

---

### Appendix D - References

The following references are **not** in alphabetical order. They are in the order they appear in the document.

- [1] Kalin, Sari. *Calculating ROI*. CIO Magazine. 15 Aug 2002. Retrieved on 4 Aug 2004 from: [www.cio.com/archive/081502/roi.html](http://www.cio.com/archive/081502/roi.html).
- [2] Rumsfeld, Donald. (9 Aug 2001). Town Hall Meeting – News Transcript. Retrieved on 17 Feb 2005 from: [www.defenselink.mil/transcripts/2001/t08092001\\_t809town.html](http://www.defenselink.mil/transcripts/2001/t08092001_t809town.html).
- [3] *Rice Development Process and Procedures*. Department of Defense: [www.eitoolkit.com](http://www.eitoolkit.com). 3 Jul 2003.
- [3] Data Mirror Corporation. *Benefits of Transformational Data Integration*. [www.eitoolkit.com](http://www.eitoolkit.com).
- [5] Jacobsen, Curt. *A Framework for Continuous Improvement in Enterprise Application Integration*. Capgemini. Not Published.
- [6] Bernard, Scott A. *An Introduction to Enterprise Architecture*. Indiana: AuthorHouse, 2004.
- [7] Buede, Dennis M. *The Engineering Design of Systems*. New York: Wiley Inter-Science, 2000.
- [8] Myerson, Judith. *Enterprise Systems Integration*. New York: Auerbach Publications, 2002.
- [9] Joint Financial Management Improvement Program. *Forum Highlights: Keys to Successful Integration/Interoperability of Business Management Systems*. 27 May 2004.
- [10] Newman, Michael. *1904 Baltimore Fire Made Standards a Hot Issue*. NIST Tech Beat. Retrieved on 10 Dec 2004 from: [http://www.nist.gov/public\\_affairs/techbeat/tb2001\\_02.htm#Centennial](http://www.nist.gov/public_affairs/techbeat/tb2001_02.htm#Centennial).
- [11] Wikipedia. Retrieved on 11 Jan 2005 from: [www.wikipedia.com](http://www.wikipedia.com).
- [12] *DoD eBusiness*. Department of Defense – Defense Information Systems Agency. Retrieved on 7 Jan 2004 from: [www.disa.mil/main/prodsol/ebusiness.html](http://www.disa.mil/main/prodsol/ebusiness.html).
- [13] Belliappa, Goutham. *Base-lining Interface Architecture for an Existing System*. Cap Gemini Ernst & Young. 2003.
- [14] Altarum. *Economic and Business Case Analysis Approach*. Not published. 14 Oct 2003.
- [15] Society of Cost Estimating and Analysis (SCEA). *Software Cost Estimating, Unit IV, Module 12*. Not published, class materials.
- [16] Enterprise Software Initiative – Department of Defense. *BearingPoint – ESI ERP Systems Integration Services (Blanket Purchase Agreement)*. Retrieved on 28 Dec 2004 from: [www.don-imit.navy.mil/esi](http://www.don-imit.navy.mil/esi).
- [17] Enterprise Software Initiative – Department of Defense. *Deloitte – ESI ERP Systems Integration Services (Blanket Purchase Agreement)*. Retrieved on 28 Dec 2004 from: [www.don-imit.navy.mil/esi](http://www.don-imit.navy.mil/esi).

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

---

- [18] von Bernewitz, Carla. Director, Army Enterprise Integration Oversight Office. Personal Interview on 18 Aug 2004.
- [19] Derby, Paul. Enterprise Architect, U.S. Army, Army Architecture Integration Cell (AAIC). Personal Interview on 11 Aug 2004.
- [20] Nyere, John, Enterprise Architect, Office of the Under Secretary of Defense (Acquisition, Technology, & Logistics). Personal Interview on 26 Aug 2004.
- [21] Wright, Larry. Senior Manager, Capgemini. Personal Interview on 14 Sept 2004.
- [22] Defense Appropriations Act of 2004, Public Law 108-87, Section 8084(b)(1). Retrieved on 3 Jan 2004 from:  
[www.dtra.mil/press\\_resources/publications/deskbook/full\\_text/Public\\_Laws/Pub.%20L.%20108-87.pdf](http://www.dtra.mil/press_resources/publications/deskbook/full_text/Public_Laws/Pub.%20L.%20108-87.pdf)
- [23] Wolfowitz, Paul, Deputy Secretary of Defense. Memo: *Information Technology Portfolio Management*. Retrieved on 13 Jan 2005 from:  
[www.army.mil/aeioo/toolkits/documents/refs/DepSecD\\_IT%20PfM\\_3-22-04.pdf](http://www.army.mil/aeioo/toolkits/documents/refs/DepSecD_IT%20PfM_3-22-04.pdf)
- [24] Serain, Daniel. *Middleware and Enterprise Application Integration*. New York: Springer, 2001.
- [25] Chappell, David A. *Enterprise Service Bus*. California; O'Reilly, 2004.
- [26] IONA. *Web Services – Setting the Stage*. [www.iona.com](http://www.iona.com).
- [27] Strauss, H. *Defense CIOs Must prepare Now for Network-Centric Operations*. Gartner (G00123410). 17 Nov 2004.
- [28] Sheehan, Jack, PM Knowledge Integration, Defense Modeling & Simulation Office. *Data Provisioning Using Authoritative Data Sources*. NDIA SBA Conference briefing, 16 May 2001. Retrieved on 13 Sept 2004 from: [www.dtic.mil/ndia/2001sbac/sheehan.pdf](http://www.dtic.mil/ndia/2001sbac/sheehan.pdf)
- [29] Heller, Roger. *An Introduction to Function Point Analysis*. Q/P Management Group, Inc. Retrieved on 12 Aug 2004 from: [www.qpmg.com/fp-intro.htm](http://www.qpmg.com/fp-intro.htm)
- [30] Longstreet Consulting, Inc. *Fundamentals of FPA*. Blue Springs, MO. Retrieved on 17 Aug 2004 from: [www.ifpug.com/fpafund.htm](http://www.ifpug.com/fpafund.htm)
- [31] *Establishment of the Army Enterprise Integration Oversight Office (AEIOO)*, Secretary of the Army. 16 Apr 2003.
- [32] *Federal Enterprise Architecture - The Technical Reference Model*. Office of Management and Budget - FEA Program Management Office (FEAPMO). August 2003.
- [33] Stenbit, John P. *Department of Defense Net-Centric Data Strategy*. Department of Defense, Chief Information Officer. 9 May 2003.
- [34] Federal CIO Council Best Practices Committee. *A Summary of First Practices and Lessons Learned in Information Technology Portfolio Management*. March 2002. Retrieved on 10 Jan 2004 from: [www.army.mil/aeioo/toolkits/documents/refs/DepSecD\\_IT%20PfM\\_3-22-04.pdf](http://www.army.mil/aeioo/toolkits/documents/refs/DepSecD_IT%20PfM_3-22-04.pdf)

## Interfaces for Enterprise Solutions

### Army Enterprise Integration Oversight Office (AEIOO)

---

- [35] Architecture and Infrastructure Committee, Federal Chief Information Officers Council. *Service Component-Based Architectures Version 2.0*. June 2004. Retrieved on 24 Dec 2004 from:  
[www.cio.gov/documents/CIOC\\_AIC\\_Service%20Component%20Based%20Architectures%20\\_2.0\\_FINAL.pdf](http://www.cio.gov/documents/CIOC_AIC_Service%20Component%20Based%20Architectures%20_2.0_FINAL.pdf)
  
- [36] Microsoft Developer Network. *Transformation vs. Translation*. Retrieved on 25 Jan 2005 from:  
[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdk/htm/ebiz\\_prog\\_map\\_edzn.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdk/htm/ebiz_prog_map_edzn.asp)